Our Ref.: 723-969

U.S. PATENT APPLICATION

Inventor(s): Dan SHIMIZU
Ko SHIOTA
Munehito OIRA
Kazuo KOSHIMA

Invention: CONTROLLER INTERFACE FOR A GRAPHICS SYSTEM

NIXON & VANDERHYE P.C.
ATTORNEYS AT LAW
1100 NORTH GLEBE ROAD
8TH FLOOR
ARLINGTON, VIRGINIA 22201-4714
(703) 816-4000
Facsimile (703) 816-4100

SPECIFICATION

Specification

Controller Interface For a Graphics System

Related Applications

This application claims priority from provisional Application No. 60/226,885, filed August 23, 2000, the contents of which are incorporated herein.

This application is related to the following applications identified below, which focus on various aspects of the graphics processing described herein. Each of the following applications are hereby incorporated herein by reference.

- provisional Application No. 60/161,915, filed October 28, 1999 and its corresponding utility Application No. 09/465,754, filed December 17, 1999, both entitled "Vertex Cache For 3D Computer Graphics",
- provisional Application No. 60/226,912, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-959), both entitled "Method and Apparatus for Buffering Graphics Data in a Graphics System ",
- provisional Application No. 60/226,889, filed August 23, 2000 and its corresponding utility Application No _____, filed _____ (atty. dkt. no. 723-958), both entitled "Graphics Pipeline Token Synchronization",
- provisional Application No. 60/226,891, filed August 23, 2000 and its corresponding utility Application No _____ , filed _____ (atty. dkt. no.

723-961), both entitled "Method And Apparatus For Direct and Indirect Texture Processing In A Graphics System",

- provisional Application No. 60/226,888, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-968), both entitled "Recirculating Shade Tree Blender For A Graphics System",
- provisional Application No. 60/226,892, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-960), both entitled "Method And Apparatus For Efficient Generation Of Texture Coordinate Displacements For Implementing Emboss-Style Bump Mapping In A Graphics Rendering System",
- provisional Application No. 60/226,893, filed August 23, 2000 and its corresponding utility Application No. _____ filed _____ (atty. dkt. no. 723-962), both entitled "Method And Apparatus For Environment-Mapped Bump-Mapping In A Graphics System",
- provisional Application No. 60/227,007, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-967), both entitled "Achromatic Lighting in a Graphics System and Method",

- provisional Application No. 60/226,900, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-964), both entitled "Method And Apparatus For Anti-Aliasing In A Graphics System",
- provisional Application No. 60/226,910, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-957), both entitled "Graphics System With Embedded Frame Buffer Having Reconfigurable Pixel Formats",
- utility Application No. 09/585,329, filed June 2, 2000, entitled "Variable Bit Field Color Encoding" (atty. dkt. no. 723-749),
- provisional Application No. 60/226,890, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-956), both entitled "Method And Apparatus For Dynamically Reconfiguring The Order Of Hidden Surface Processing Based On Rendering Mode",
- provisional Application No. 60/226,915, filed August 23, 2000 and its corresponding utility Application No. _____ filed _____ (atty. dkt. no. 723-973), both entitled "Method And Apparatus For Providing Non-Photorealistic Cartoon Outlining Within A Graphics System",

- provisional Application No. 60/227,032, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____, (atty. dkt. no. 723-954), both entitled "Method And Apparatus For Providing Improved Fog Effects In A Graphics System",
- provisional Application No. 60/227,033, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-955), both entitled "Method And Apparatus For Texture Tiling In A Graphics System",
- provisional Application No. 60/226,899, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-971), both entitled "Method And Apparatus For Pre-Caching Data In Audio Memory",
- provisional Application No. 60/226,913, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-965), both entitled "Z-Texturing",
- provisional Application No. 60/227,031, filed August 23, 2000 entitled "Application Program Interface for a Graphics System" (atty. dkt. no. 723-880),
- provisional Application No. 60/227,030, filed August 23, 2000 and its corresponding utility Application No. _____ , filed _____ (atty. dkt. no.

723-963), both entitled "Graphics System With Copy Out Conversions Between Embedded Frame Buffer And Main Memory",

- provisional Application No. 60/226,886, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-970), both entitled "Method and Apparatus for Accessing Shared Resources",
- provisional Application No. 60/226,884, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-972), both entitled "External Interfaces For A 3D Graphics and Audio Coprocessor",
- provisional Application No. 60/226,894, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-974), both entitled "Graphics Processing System With Enhanced Memory Controller",
- provisional Application No. 60/226,914, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____, (atty. dkt. no. 723-966), both entitled " Low Cost Graphics System With Stitching Hardware Support For Skeletal Animation", and

- provisional Application No. 60/227,006, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-953), both entitled " Shadow Mapping In A Low Cost Graphics System".

Field of the Invention

The present invention relates to computer graphics, and more particularly to interactive graphics systems such as home video game platforms. Still more particularly this invention relates to a controller interface used to connect a controller to a graphics system.

Background And Summary Of The Invention

Many of us have seen films containing remarkably realistic dinosaurs, aliens, animated toys and other fanciful creatures. Such animations are made possible by computer graphics. Using such techniques, a computer graphics artist can specify how each object should look and how it should change in appearance over time, and a computer then models the objects and displays them on a display such as your television or a computer screen. The computer takes care of performing the many tasks required to make sure that each part of the displayed image is colored and shaped just right based on the position and orientation of each object in a scene, the direction in which light seems to strike each object, the surface texture of each object, and other factors.

Because computer graphics generation is complex, computer-generated three-dimensional graphics just a few years ago were mostly limited to expensive specialized flight simulators, high-end graphics workstations and supercomputers. The public saw some of the images generated by these computer systems in movies and expensive television advertisements, but most of us couldn't actually interact

with the computers doing the graphics generation. All this has changed with the availability of relatively inexpensive 3D graphics platforms such as, for example, the Nintendo 64® and various 3D graphics cards now available for personal computers. It is now possible to interact with exciting 3D animations and simulations on relatively inexpensive computer graphics systems in your home or office.

In an interactive real time system such as a gaming platform, user-manipulable controls are provided for player inputs. The controls may take many forms including buttons, switches, joysticks, trackballs and the like. These player inputs are used during game play, for example, to move characters left, right, up and down; to take character actions such as jumping; etc. Thus, in game systems, some means must be provided to interface with user-manipulable controls such as hand-held controllers or the like. The present invention provides such an interface. More specifically, the present invention provides an interface for interfacing an audio and graphics coprocessor with a variety of different types of accessory devices including but not limited to hand-held game controllers. The serial interface provides a single bit serial interface using a state-based interface protocol. The interface supports four separate serial interfaces to four hand-held controllers or associated devices. Each interface can be accessed in parallel. In a controller mode, the last state of the controller is stored in a double-buffered register to support simple main processor reads for determining state. The example embodiment automatically polls controller state using hardware circuitry with configurable polling periods. A bulk mode supports changeable data size. A pair of light gun signals can be used to control separate horizontal/vertical counters to support flash and shutter light guns. An LCD shutter can be supported through automatic polling and a serial control command. The system interface includes

automatic control of presence detect to save effort on the part of the main processor.

In accordance with one particular aspect of the present invention, a video game system includes a game program executing system executing a game program and one or more controllers supplying user inputs to the game program executing system. An interface between the controllers and the game program executing system is programmable to periodically poll the controller without involvement of the game program executing system.

Brief Description Of The Drawings

These and other features and advantages provided by the invention will be better and more completely understood by referring to the following detailed description of presently preferred embodiments in conjunction with the drawings, of which:

Figure 1 is an overall view of an example interactive computer graphics system;

Figure 2 is a block diagram of the Figure 1 example computer graphics system;

Figure 3 is a block diagram of the example graphics and audio processor shown in Figure 2;

Figure 4 is a block diagram of the example 3D graphics processor shown in Figure 3;

Figure 5 is an example logical flow diagram of the Figure 4 graphics and audio processor;

Figure 6 shows an example peripheral controller 162;

Figures 7A and 7B show an example serial interface;

Figure 7C shows example serial interface registers;

Figure 8 shows a more detailed view of the serial interface of Figures 7A and 7B;

Figure 9 shows a more detailed view of a portion of the serial interface shown in Figure 8;

Figure 10 shows an example controller;

Figure 11 shows the example system elements involved in data transfers using a communication RAM;

Figure 12 is used to illustrate the meaning of the X line interval register and the Y times register;

Figure 13 shows an example light gun;

Figures 14A-14C are timing diagrams for explaining the light gun and light pen operations;

Figures 15A-15E depict signals used in data transfer;

Figure 16 illustrates the communication protocol between the serial interface and the controllers;

Figure 17 depicts the lines for the channels of serial interface 1000; and

Figures 18A and 18B show example alternative compatible implementations.

Detailed Description Of Example Embodiments Of The Invention

Figure 1 shows an example interactive 3D computer graphics system 50. System 50 can be used to play interactive 3D video games with interesting stereo sound. It can also be used for a variety of other applications.

In this example, system 50 is capable of processing, interactively in real time, a digital representation or model of a three-dimensional world. System 50 can display some or all of the world from any arbitrary viewpoint. For example, system 50 can interactively change the viewpoint in response to real time inputs from handheld controllers 52a, 52b or other input devices. This allows the game player to see the world through the eyes of someone within or outside of the world. System 50 can be used for applications that do not require real time 3D interactive display (e.g., 2D display generation and/or non-interactive display), but the capability of displaying quality 3D images very quickly can be used to create very realistic and exciting game play or other graphical interactions.

To play a video game or other application using system 50, the user first connects a main unit 54 to his or her color television set 56 or other display device by connecting a cable 58 between the two. Main unit 54 produces both video signals and audio signals for controlling color television set 56. The video signals are what controls the images displayed on the television screen 59, and the audio signals are played back as sound through television stereo loudspeakers 61L, 61R.

The user also needs to connect main unit 54 to a power source. This power source may be a conventional AC adapter (not shown) that plugs into a standard home electrical wall socket and converts the house current into a lower DC voltage signal suitable for powering the main unit 54. Batteries could be used in other implementations.

The user may use hand controllers 52a, 52b to control main unit 54. Controls 60 can be used, for example, to specify the direction (up or down, left or right, closer or further away) that a character displayed on television 56 should move within a 3D world. Controls 60 also provide input for other applications (e.g., menu selection, pointer/cursor control, etc.). Controllers 52 can take a variety of forms. In this example, controllers 52 shown each include controls 60 such as joysticks, push buttons and/or directional switches. Controllers 52 may be connected to main unit 54 by cables or wirelessly via electromagnetic (e.g., radio or infrared) waves.

To play an application such as a game, the user selects an appropriate storage medium 62 storing the video game or other application he or she wants to play, and inserts that storage medium into a slot 64 in main unit 54. Storage medium 62 may, for example, be a specially encoded and/or encrypted optical and/or magnetic disk. The user may operate a power switch 66 to turn on main unit 54 and cause the main unit to begin running the video game or other application based on the software stored in the storage medium 62. The user may operate controllers 52 to provide inputs to main unit 54. For example, operating a control 60 may cause the game or other application to start. Moving other controls 60 can cause animated characters to move in different directions or change the user's point of view in a 3D world. Depending upon the particular software stored within the storage medium 62, the various controls 60 on the controller 52 can perform different functions at different times.

Example Electronics of Overall System

Figure 2 shows a block diagram of example components of system 50. The primary components include:

- a main processor (CPU) 110,
- a main memory 112, and
- a graphics and audio processor 114.

INS A1 In this example, main processor 110 (e.g., an enhanced IBM Power PC 750) receives inputs from handheld controllers 108 (and/or other input devices) via graphics and audio processor 114. Main processor 110 interactively responds to user inputs, and executes a video game or other program supplied, for example, by external storage media 62 via a mass storage access device 106 such as an optical disk drive. As one example, in the context of video game play, main processor 110 can perform collision detection and animation processing in addition to a variety of interactive and control functions.

In this example, main processor 110 generates 3D graphics and audio commands and sends them to graphics and audio processor 114. The graphics and audio processor 114 processes these commands to generate interesting visual images on display 59 and interesting stereo sound on stereo loudspeakers 61R, 61L or other suitable sound-generating devices.

Example system 50 includes a video encoder 120 that receives image signals from graphics and audio processor 114 and converts the image signals into analog and/or digital video signals suitable for display on a standard display device such as a computer monitor or home color television set 56. System 50 also includes an audio codec (compressor/decompressor) 122 that compresses and decompresses digitized audio signals and may also convert between digital and analog audio signaling formats as needed. Audio codec 122 can receive audio inputs via a buffer 124 and provide them to graphics and audio processor 114 for processing (e.g., mixing with other audio signals the processor generates and/or receives via a streaming audio output of mass storage access device 106). Graphics and audio

processor 114 in this example can store audio related information in an audio memory 126 that is available for audio tasks. Graphics and audio processor 114 provides the resulting audio output signals to audio codec 122 for decompression and conversion to analog signals (e.g., via buffer amplifiers 128L, 128R) so they can be reproduced by loudspeakers 61L, 61R.

INS A2 Graphics and audio processor 114 has the ability to communicate with various additional devices that may be present within system 50. For example, a parallel digital bus 130 may be used to communicate with mass storage access device 106 and/or other components. A serial peripheral bus 132 may communicate with a variety of peripheral or other devices including, for example:

- a programmable read-only memory and/or real time clock 134,
- a modem 136 or other networking interface (which may in turn connect system 50 to a telecommunications network 138 such as the Internet or other digital network from/to which program instructions and/or data can be downloaded or uploaded), and
- flash memory 140.

A further external serial bus 142 may be used to communicate with additional expansion memory 144 (e.g., a memory card) or other devices. Connectors may be used to connect various devices to busses 130, 132, 142.

Example Graphics And Audio Processor

INS A3 Figure 3 is a block diagram of an example graphics and audio processor 114. Graphics and audio processor 114 in one example may be a single-chip ASIC (application specific integrated circuit). In this example, graphics and audio processor 114 includes:

- a processor interface 150,

- a memory interface/controller 152,
- a 3D graphics processor 154,
- an audio digital signal processor (DSP) 156,
- an audio memory interface 158,
- an audio interface and mixer 160,
- a peripheral controller 162, and
- a display controller 164.

3D graphics processor 154 performs graphics processing tasks. Audio digital signal processor 156 performs audio processing tasks. Display controller 164 accesses image information from main memory 112 and provides it to video encoder 120 for display on display device 56. Audio interface and mixer 160 interfaces with audio codec 122, and can also mix audio from different sources (e.g., streaming audio from mass storage access device 106, the output of audio DSP 156, and external audio input received via audio codec 122). Processor interface 150 provides a data and control interface between main processor 110 and graphics and audio processor 114.

Memory interface 152 provides a data and control interface between graphics and audio processor 114 and memory 112. In this example, main processor 110 accesses main memory 112 via processor interface 150 and memory interface 152 that are part of graphics and audio processor 114. Peripheral controller 162 provides a data and control interface between graphics and audio processor 114 and the various peripherals mentioned above. Audio memory interface 158 provides an interface with audio memory 126.

Example Graphics Pipeline

Figure 4 shows a more detailed view of an example 3D graphics processor 154. 3D graphics processor 154 includes, among other things, a command processor 200 and a 3D graphics pipeline 180. Main processor 110 communicates streams of data (e.g., graphics command streams and display lists) to command processor 200. Main processor 110 has a two-level cache 115 to minimize memory latency, and also has a write-gathering buffer 111 for uncached data streams targeted for the graphics and audio processor 114. The write-gathering buffer 111 collects partial cache lines into full cache lines and sends the data out to the graphics and audio processor 114 one cache line at a time for maximum bus usage.

Command processor 200 receives display commands from main processor 110 and parses them -- obtaining any additional data necessary to process them from shared memory 112. The command processor 200 provides a stream of vertex commands to graphics pipeline 180 for 2D and/or 3D processing and rendering. Graphics pipeline 180 generates images based on these commands. The resulting image information may be transferred to main memory 112 for access by display controller/video interface unit 164 -- which displays the frame buffer output of pipeline 180 on display 56.

Figure 5 is a logical flow diagram of graphics processor 154. Main processor 110 may store graphics command streams 210, display lists 212 and vertex arrays 214 in main memory 112, and pass pointers to command processor 200 via bus interface 150. The main processor 110 stores graphics commands in one or more graphics first-in-first-out (FIFO) buffers 210 it allocates in main memory 110. The command processor 200 fetches:

- command streams from main memory 112 via an on-chip FIFO memory buffer 216 that receives and buffers the graphics commands for synchronization/flow control and load balancing,
- display lists 212 from main memory 112 via an on-chip call FIFO memory buffer 218, and
- vertex attributes from the command stream and/or from vertex arrays 214 in main memory 112 via a vertex cache 220.

Command processor 200 performs command processing operations 200a that convert attribute types to floating point format, and pass the resulting complete vertex polygon data to graphics pipeline 180 for rendering/rasterization. A programmable memory arbitration circuitry 130 (see Figure 4) arbitrates access to shared main memory 112 between graphics pipeline 180, command processor 200 and display controller/video interface unit 164.

Figure 4 shows that graphics pipeline 180 may include:

- a transform unit 300,
- a setup/rasterizer 400,
- a texture unit 500,
- a texture environment unit 600, and
- a pixel engine 700.

Transform unit 300 performs a variety of 2D and 3D transform and other operations 300a (see Figure 5). Transform unit 300 may include one or more matrix memories 300b for storing matrices used in transformation processing 300a. Transform unit 300 transforms incoming geometry per vertex from object space to screen space; and transforms incoming texture coordinates and computes projective texture coordinates (300c). Transform unit 300 may also perform

polygon clipping/culling 300d. Lighting processing 300e also performed by transform unit 300b provides per vertex lighting computations for up to eight independent lights in one example embodiment. Transform unit 300 can also perform texture coordinate generation (300c) for embossed type bump mapping effects, as well as polygon clipping/culling operations (300d).

Setup/rasterizer 400 includes a setup unit which receives vertex data from transform unit 300 and sends triangle setup information to one or more rasterizer units (400b) performing edge rasterization, texture coordinate rasterization and color rasterization.

Texture unit 500 (which may include an on-chip texture memory (TMEM) 502) performs various tasks related to texturing including for example:

- retrieving textures 504 from main memory 112,
- texture processing (500a) including, for example, multi-texture handling, post-cache texture decompression, texture filtering, embossing, shadows and lighting through the use of projective textures, and BLIT with alpha transparency and depth,
- bump map processing for computing texture coordinate displacements for bump mapping, pseudo texture and texture tiling effects (500b), and
- indirect texture processing (500c).

Texture unit 500 outputs filtered texture values to the texture environment unit 600 for texture environment processing (600a). Texture environment unit 600 blends polygon and texture color/alpha/depth, and can also perform texture fog processing (600b) to achieve inverse range based fog effects. Texture environment unit 600 can provide multiple stages to perform a variety of other interesting

environment-related functions based for example on color/alpha modulation, embossing, detail texturing, texture swapping, clamping, and depth blending..

INS A6 Pixel engine 700 performs depth (z) compare (700a) and pixel blending (700b). In this example, pixel engine 700 stores data into an embedded (on-chip) frame buffer memory 702. Graphics pipeline 180 may include one or more embedded DRAM memories 702 to store frame buffer and/or texture information locally. Z compares 700a' can also be performed at an earlier stage in the graphics pipeline 180 depending on the rendering mode currently in effect (e.g., z compares can be performed earlier if alpha blending is not required). The pixel engine 700 includes a copy operation 700c that periodically writes on-chip frame buffer 702 to main memory 112 for access by display/video interface unit 164. This copy operation 700c can also be used to copy embedded frame buffer 702 contents to textures in the main memory 112 for dynamic texture synthesis effects. Anti-aliasing and other filtering can be performed during the copy-out operation. The frame buffer output of graphics pipeline 180 (which is ultimately stored in main memory 112) is read each frame by display/video interface unit 164. Display controller/video interface 164 provides digital RGB pixel values for display on display 56.

Example Peripheral Controller

INS A7 Figure 6 shows an example peripheral controller 162. In this example, peripheral controller 162 includes a serial interface 1000, an external interface 1100, a disk interface 1200 and an audio interface 1300. Serial interface 1000 is used to communicate with controllers 52 or other devices that can be coupled to one of four serial ports of system 50. External interface 1100 is used to communicate with a variety of devices such as PROM RTC 134, modem 136, flash memory 140, memory card 144, etc. via various buses 132, 142. Disk interface

1200 is used to communicate with mass storage access device 106 via a parallel bus 130. Audio interface 1300 is used to stream the audio output data from an audio buffer in main memory 112 to audio codec 122.

In the example embodiment, the external interface 1100 and disk interface 1200 have direct access to memory controller 152 via a bus 900. Details of the operation of memory controller 152 may be found in provisional Application No. 60/226,894, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-974), both entitled "Graphics Processing System with Enhanced Memory Controller" and provisional Application No. 60/226,886, filed August 23, 2000 and its corresponding utility Application No. _____, filed _____ (atty. dkt. no. 723-970), both entitled "Method and Apparatus for Accessing Shared Resources." The contents of each of these applications are incorporated herein by reference. In addition, each one of interfaces 1000, 1100, 1200 and 1300 as well as audio digital signal processor 156 share a common bus 902 used to communicate between these components and a bus interface 904. The bus interface 904, in turn, can be used to arbitrate access to graphics unit 180. In the example embodiment, there is also a connection 906 between DSP 156 and audio interface 1300.

Briefly, disk interface 1200 provides an interface to mass storage access device 106 providing a direct memory access (DMA) capability with interrupt. Serial interface 1000 provides a serial interface to hand controllers 52 or other serial devices using automatic controller polling and bulk data mode. The serial interface also includes a light gun interface. The external interface 1100 provides multiple SBI buses as well as a memory-mapped area for boot PROM 134. Audio interface 1300 provides an output to audio codec 122 as well as an input for streaming audio from mass storage access device 106.

Example Serial Interface

Figures 7A and 7B show an example serial interface 1000. In this particular example, serial interface 1000 is a single bit serial interface that runs at 250 kHz. This single bit serial interface is similar to the controller interface used in the prior art Nintendo 64® product manufactured by Nintendo, but there are some differences. Example serial interface 1000 provides the following features in the example embodiment:

- four separate 250 kHz serial interfaces for four controllers 52,
- each interface can be accessed in parallel,
- in controller mode, the last state of the controller 52 is in a double-buffered processor input/output register so that main processor 110 can simply read the register to determine the controller state,
- the controller state is automatically polled by hardware with configurable polling periods,
- bulk mode (changeable data size),
- two light gun signals are used to control two separate horizontal/vertical counters to support both flash and shutter light guns,
- an LCD shutter is supported through automatic polling and serial control commands, and
- the serial interface 1000 can automatically detect the presence of hand controllers 52.

Figure 7A shows the external interface of serial interface 1000. In this example, there are four separate controller ports 1002 on system 50. Each port 1002 has a pair of input and output pins (shown by the "x" mark blocks in Figure 7A). The input pin connects directly to an external game controller 52 in the example embodiment. The output pin in the example embodiment connects to an

external open-drain driver (see Figure 9) which in turn connects directly to the external game controller 52. In the example embodiment, two of the ports 1002 have horizontal/vertical latch signals that can be used to latch horizontal/vertical counters within the video interface 164. These signals combined with the functionality of serial interface 1000 provide support for flash and shutter type light guns. The vertical latch and control registers used for this functionality are located in the video interface 164 in the example embodiment. Figure 7A shows each of the four serial ports 1002 including an SIDI (bi-directional) line and an SIDO (uni-directional) controller output-to-serial interface line. The following shows example descriptions of these two signals:

Name	Dir	Type	Description
SIDI[3:0]	I	LVCMOS	Serial Interface Data Input: SIDI[3:0] are input signals, each bit is a separate half-duplex, 250kbit/s input serial channel. The serial protocol is an asynchronous interface and is self timed, using a pulse width modulated signaling scheme.
SID0[3:0]	O	LVCMOS	Serial Interface Data Output: SIDO[3:0] are output signals, each bit is a separate half-duplex, 250kbit/s output serial channel. The serial protocol is an asynchronous interface and is self timed, using a pulse width modulated signaling scheme.

Figure 7B is a more detailed block diagram of serial interface 1000. As shown in this Figure, serial interface 1000 includes a main processor interface 1010, serial interface communication circuitry and registers 1012, a small (128 byte) communication RAM 1014, and an input/output buffer arrangement 1016 for each of the four serial ports 1002.

Figure 7C shows an example set of registers (register map) used to control serial interface 1000 in the example embodiment. The base address for these serial interface registers in the example embodiment is 0x0C006400. The following describes each of these various example registers in the example embodiment:

SIC0OUTBUF SI Channel 0 Output Buffer

Mnemonic: SIC0OUTBUF
Offset: 0x00
Size 32 bits

SIC0OUTBUF				
Bits	Mnemonic	Type	Reset	Description
31..24		R	0x0	Reserved
23..16	CMD	RW	0x0	Command: This byte is the opcode for the command sent to the controller during each command/response packet. This is the first data byte sent from the SI I/F to the game controller in the command/response packet.
15..8	OUTPUT0	RW	0x0	Output Byte 0: This is the first data byte of the command packet. It is the second data byte sent from the SI I/F to the game controller in the command/response packet.
7..0	OUTPUT1	RW	0x0	Output Byte 1: This is the second data byte of the command packet. It is the third data byte sent from the SI I/F to the game controller in the command/response packet.

This register is double buffered, so main processor writes to the SIC0OUTBUF will not interfere with the serial interface output transfer. Internally, a second buffer is used to hold the output data to be transferred across the serial interface. To check if SIC0OUTBUF has been transferred to the second buffer, main processor 110 polls the SISR[WRST0] register. When SIC0OUTBUF is transferred, SISR[WRST0] is cleared.

SIC0INBUF SI Channel 0 Input Buffer High

Mnemonic: SIC0INBUFH
Offset: 0x04
Size 32 bits

SIC0INBUFH				
Bits	Mnemonic	Type	Reset	Description
31	ERRSTAT	R	0x0	Error Status: This bit represents the current error status for the last SI polling transfer on channel 0. This register is updated after each polling transfer on this channel. 0 = No error on last transfer 1 = Error on last transfer
30	ERRLATCH	R	0x0	Error Latch: This bit is an error status summary of the SISR error bits for this channel. If an error has occurred on a past SI transfer on channel 0 (polling or Com transfer), this bit will be set. To determine the exact error, read the SISR register. This bit is actually an 'or' of the latched error status bits for channel 0 in the SISR. The bit is cleared by clearing the appropriate error status bits latched in the SISR. The no response error indicates that a controller is not present on this channel. 0 = No errors latched 1 = Error latched. Check SISR.
29..24	INPUT0	R	0x0	Input Byte 0: This is the first data byte of the response packet sent from the game controller to the SI I/F for channel 0. The top two bits of the byte returning from the controller are assumed to be '0', so they are not included.
23..16	INPUT1	R	0x0	Input Byte 1: This is the second data byte of the response packet sent from the game controller to the SI I/F for channel 0.
15..8	INPUT2	R	0x0	Input Byte 2: This is the third data byte of the response packet sent from the game controllers to the SI I/F for channel 0.
7:0	INPUT3	R	0x0	Input Byte 3: This is the fourth data byte of the response packet sent from the game controller to the SI I/F for channel 0.

5

SIC0INBUFH and SIC0INBUFL are double buffered to prevent inconsistent data reads due to the main processor 110 conflicting with incoming serial interface data. To insure data read from SIC0INBUFH and SIC0INFUBL are consistent, a

locking mechanism prevents the double buffer from copying new data to these registers. Once SIC0INBUFH is read, both SIC0INBUFH and SIC0INBUFL are 'locked' until SIC0INBUFL is read. While the buffers are 'locked', new data is not copied into the buffers. When SIC0INBUFL is read, the buffers become unlocked again.

SIC0INBUF SI Channel 0 Input Buffer Low

Mnemonic: SIC0INBUFL
Offset: 0x08
Size 32 bits

SIC0INBUFL				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31..24	INPUT4	R	0x0	Input Byte 4: See Description of SIC1INBUFH[INPUT1].
23..16	INPUT5	R	0x0	Input Byte 5: See Description of SIC1INBUFH[INPUT1].
15..8	INPUT6	R	0x0	Input Byte 6: See Description of SIC1INBUFH[INPUT1].
7..0	INPUT7	R	0x0	Input Byte 7: See Description of SIC1INBUFH[INPUT1].

SIC1OUTBUF SI Channel 1 Output Buffer

Mnemonic: SIC1OUTBUF
Offset: 0x0C
Size 32 bits

SIC1OUTBUF				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31..24		R	0x0	Reserved
23..16	CMD	RW	0x0	Command: For SI channel 1. See SIC0OUTBUFF[CMD] description.
15..8	OUTPUT0	RW	0x0	Output Byte 0: For SI channel 1. See SIC0OUTBUFF[OUTPUT0] description.
7..0	OUTPUT1	RW	0x0	Output Byte 1: For SI channel 1. See SIC0OUTBUFF[OUTPUT1] description.

SIC1INBUF SI Channel 1 Input Buffer High

Mnemonic: SIC1INBUFH
Offset: 0x10
Size 32 bits

SIC1INBUFH				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31	ERRSTAT	R	0x0	Error Status: See Description of SIC0INBUFH[ERRSTAT].
30	ERRLATCH	R	0x0	Error Latch: See Description of SIC0INBUFH[ERRLATCH].
29..24	INPUT0	R	0x0	Input Byte 0: See Description of SIC0INBUFH[INPUT0].
23..16	INPUT1	R	0x0	Input Byte 1: See Description of SIC0INBUFH[INPUT1].
15..8	INPUT2	R	0x0	Input Byte 2: See Description of SIC0INBUFH[INPUT1].
7..0	INPUT3	R	0x0	Input Byte 3: See Description of SIC0INBUFH[INPUT1].

SIC1INBUF SI Channel 1 Input Buffer Low

Mnemonic: SIC1INBUFL
Offset: 0x14
Size 32 bits

SIC1INBUFL				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31..24	INPUT4	R	0x0	Input Byte 4: See Description of SIC0INBUFH[INPUT1].
23..16	INPUT5	R	0x0	Input Byte 5: See Description of SIC0INBUFH[INPUT1].
15..8	INPUT6	R	0x0	Input Byte 6: See Description of SIC0INBUFH[INPUT1].
7..0	INPUT7	R	0x0	Input Byte 7: See Description of SIC0INBUFH[INPUT1].

SIC2OUTBUF SI Channel 2 Output Buffer

Mnemonic: SIC2OBUF
Offset: 0x18
Size 32 bits

SIC2OUTBUF				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31..24		R	0x0	Reserved
23..16	CMD	RW	0x0	Command: For SI Channel 2. See SIC0OUTBUFF[CMD] description
15..8	OUTPUT0	RW	0x0	Output Byte 0: For SI channel 2. See SIC0OUTBUFF[OUTPUT0] description.
7..0	OUTPUT1	RW	0x0	Output Byte 1: For SI channel 2. See SIC0OUTBUFF[OUTPUT1] description.

SIC2INBUF SI Channel 2 Input Buffer High

Mnemonic: SIC2INBUFH
Offset: 0x1C
Size 32 bits

SIC2INBUFH				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31	ERRSTAT	R	0x0	Error Status: See Description of SIC0INBUFH[ERRSTAT].
30	ERRLATCH	R	0x0	Error Latch: See Description of SIC0INBUFH[ERRLATCH].
29..24	INPUT0	R	0x0	Input Byte 0: See Description of SIC0INBUFH[INPUT1]..
23..16	INPUT1	R	0x0	Input Byte 1: See Description of SIC0INBUFH[INPUT0].
15..8	INPUT2	R	0x0	Input Byte 2: See Description of SIC0INBUFH[INPUT1].
7..0	INPUT3	R	0x0	Input Byte 3: See Description of SIC0INBUFH[INPUT1].

SIC2INBUF SI Channel 2 Input Buffer Low

Mnemonic: SIC2INBUFL
Offset: 0x20
Size 32 bits

SIC2INBUFL				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31..24	INPUT4	R	0x0	Input Byte 4: See Description of SIC0INBUFH[INPUT0].

23..16	INPUT5	R	0x0	Input Byte 5: See Description of SIC0INBUFH[INPUT0].
15..8	INPUT6	R	0x0	Input Byte 6: See Description of SIC0INBUFH[INPUT0].
7..0	INPUT7	R	0x0	Input Byte 7: See Description of SIC0INBUFH[INPUT0].

SIC3OUTBUF SI Channel 3 Output Buffer

Mnemonic: SIC3OBUF
Offset: 0x24
Size 32 bits

SIC3OUTBUF				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31..24		R	0x0	Reserved
23..16	CMD	RW	0x0	Command: For SI channel 3. See SIC0OUTBUFF[CMD] description
15..8	OUTPUT0	RW	0x0	Output Byte 0: For SI channel 3. See SIC0OUTBUFF[OUTPUT0] description
7..0	OUTPUT1	RW	0x0	Output Byte 1: For SI channel 3. See SIC0OUTBUFF[OUTPUT1] description

SIC3INBUF SI Channel 3 Input Buffer High

Mnemonic: SIC3INBUFH
Offset: 0x28
Size 32 bits

SIC3INBUFH				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31	ERRSTAT	R	0x0	Error Status: See Description of SIC0INBUFH[ERRSTAT].
30	ERRLATCH	R	0x0	Error Latch: See Description of SIC0INBUFH[ERRLATCH].
29..24	INPUT0	R	0x0	Input Byte 0: See Description of SIC0INBUFH[INPUT0].
23..16	INPUT1	R	0x0	Input Byte 1: See Description of SIC0INBUFH[INPUT1].
15..8	INPUT2	R	0x0	Input Byte 2: See Description of SIC0INBUFH[INPUT1].
7..0	INPUT3	R	0x0	Input Byte 3: See Description of SIC0INBUFH[INPUT1].

SIC3INBUF SI Channel 3 Input Buffer Low

Mnemonic: SIC3INBUFL
Offset: 0x2C
Size 32 bits

SIC4INBUFL				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31..24	INPUT4	R	0x0	Input Byte 4: See Description of SIC0INBUFH[INPUT1].
23..16	INPUT5	R	0x0	Input Byte 5: See Description of SIC0INBUFH[INPUT1].
15..8	INPUT6	R	0x0	Input Byte 6: See Description of SIC0INBUFH[INPUT1].
7..0	INPUT7	R	0x0	Input Byte 7: See Description of SIC0INBUFH[INPUT1].

5

SIPOLL SI Poll Register

Mnemonic: SIPOLL
Offset: 0x30
Size 32 bits

SIPOLL				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31..26		R	0x0	Reserved
25..16	X	RW	0x07	X lines register: determines the number of horizontal video lines between polling (the polling interval). The polling begins at vsync. 0x07 is the minimum setting (determined by the time required to complete a single polling of the controller). The maximum setting depends on the current video mode (number of lines per vsync) and the SIPOLL[Y] register. This register takes affect after vsync.
15..8	Y	RW	0x0	Y times register: This register determines the number of times the SI controllers are polled in a single frame. This register takes affect after vsync.
7	EN0	RW	0x0	Enable channel 0: Enable polling of channel 0. When the channel is enabled, polling begins at the next vblank. When the channel is disabled, polling is stopped immediately after the current transaction. The status of this bit does not affect communication RAM transfers on this channel. 1 = Polling of channel 0 is enabled 0 = Polling of channel 0 is disabled
6	EN1	RW	0x0	Enable channel 1: See description for SIPOLL[EN0].

5	EN2	RW	0x0	Enable channel 2: See Description for SIPOLL[EN0].
4	EN3	RW	0x0	Enable channel 3: See Description for SIPOLL[EN0].
3	VBCPY0	RW	0x0	Vblank copy output channel 0: Normally main processor writes to the SIC0OUTBUF register are copied immediately to the channel 0 output buffer if a transfer is not currently in progress. When this bit is asserted, main processor writes to channel 0's SIC0OUTBUF will only be copied to the outbuffer on vblank. This is used to control the timing of commands to 3D LCD shutter glasses connected to the VI. 1 = Copy SIC0OUTBUF to output buffer only on vblank. 0 = Copy SIC0OUTBUF to output buffer after writing.
2	VBCPY1	RW	0x0	Vblank copy output channel 1: See Description for SIPOLL[VBCPY0].
1	VBCPY2	RW	0x0	Vblank copy output channel 2: See Description for SIPOLL[VBCPY0].
0	VBCPY3	RW	0x0	Vblank copy output channel 3: See Description for SIPOLL[VBCPY0].

SICOMCSR SI Communication Control Status Register

Mnemonic: SICOMCSR
Offset: 0x34
Size 32 bits

SICOMCSR				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31	TCINT	RWC	0x0	Transfer Complete Interrupt Status and clear. On read this bit indicates the current status of the communication transfer complete interrupt. When a '1' is written to this register, the interrupt is cleared. Write: 0 = No effect 1 = Transfer Complete Interrupt Read: 0 = Transfer Complete Interrupt not requested 1 = Transfer Complete Interrupt has been requested

30	TCINTMSK	RW	0x0	Transfer Complete Interrupt Mask: Interrupt masking prevents the interrupt from being sent to the main processor, but does not affect the assertion of SICOMCSR[TCINT] 0 = Interrupt masked 1 = Interrupt enabled
29	COMERR	R	0x0	Communication Error: This indicates whether the last SI communication transfer had an error. See SiSr for the cause of the error. 0 = No error 1 = Error on transfer
28	RDSTINT	R	0x0	Read Status Interrupt Status and clear. On read this bit indicates the current status of the Read Status interrupt. The interrupt is set whenever SISR[RDSTn] bits are set. The interrupt is cleared when all of the RdSt bits in the SISR are cleared by reading from the Si Channel Input Buffers. This interrupt can be used to indicate that a polling transfer has completed and new data is captured in the input registers Read: 0 = Transfer Complete Interrupt not requested 1 = Transfer Complete Interrupt has been requested
27	RDSTINTMSK	RW	0x0	Read Status interrupt Mask: Interrupt masking prevents the interrupt from being sent to the main processor, but does not affect the assertion of SICOMCSR[RDSTINT] 0 = Interrupt masked 1 = Interrupt enabled
26..23		R	0x0	Reserved
22..16	OUTLNGTH	RW	0x0	Communication Channel Output Length in bytes. Minimum transfer is 1 byte. A value of 0x00 will transfer 128 bytes. These bits should not be modified while SICOM transfer is in progress.
15		R	0x0	Reserved
14..8	INLNGTH	RW	0x0	Communication Channel Output Length in bytes. Minimum transfer is 1 byte. A value of 0x00 will transfer 128 bytes. These bits should not be modified while SICOM transfer is in progress.

2..1	CHANNEL	RW	0x0	Channel: determines which SI channel will be used the communication interface. 00 = Channel 1 01 = Channel 2 10 = Channel 3 11 = Channel 4 These bits should not be modified while SICOM transfer is in progress.
0	TSTART	RW	0x0	Transfer Start: When a '1' is written to this register, the current communication transfer is executed. The transfer begins immediately after the current transaction on this channel has completed. When read this bit represents the current transfer status. Once a communication transfer has been executed, polling will resume at the next vblank if the channel's SIPOLL[ENn] bit is set. Write: 0 = Do not start command 1 = Start command Read: 0 = Command Complete 1 = Command Pending

When programming the SICOMCSR after a SICOM transfer has already started (e.g., SICOMCSR[TSTART] is set), the example software reads the current value first, then and/or in the proper data and then writes the new data back. The software should not modify any of the transfer parameters (OUTLNGTH, INLNGTH, CHANNEL) until the current transfer is complete. This is done to prevent a SICOM transfer already in progress from being disturbed. When writing the data back, the software should not set the TSTART bit again unless the current transfer is complete and another transfer is required.

SISI SI Status Register

Mnemonic: SISR
Offset: 0x38
Size 32 bits

SISR				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31	WR	RW	0x0	Write SICnOUTBUF Register: This register controls and indicates whether the SICnOUTBUFs have been copied to the double-buffered output buffers. This bit is cleared after the buffers have been copied. Write 1 = Copy all buffers 0 = No effect Read 1 = Buffer not copied 0 = Buffer copied
30		R	0x0	Reserved
29	RDST0	R	0x0	Read Status SIC0OINBUF Register: This register indicates whether the SIC0INBUFs have been captured new data and whether the data has already been read by the main processor (read indicated by main processor read of SIC0INBUF[ERRSTAT, ERRLATCH, INPUT0, INPUT1)] 1 = New data available, not read by main processor 0 = No new data available, already read by main processor
28	WRST0	R	0x0	Write Status SIC0OUTBUF Register: This register indicates whether the SIC0OUTBUFs have been copied to the double buffered output buffers. This bit is cleared after the buffers have been copied. 1 = Buffer not copied 0 = Buffer copied
27	NOREP0	RWC	0x0	No Response Error Channel 0: This register indicates that a previous transfer resulted in no response from the controller. This can also be used to detect whether a controller is connected. If no controller is connected, this bit will be set. Once set this bit remains set until it is cleared by the main processor. To clear this bit write '1' to this register. Write: 0 = No effect 1 = Clear No Response Error Read: 0 = No Response Error not asserted 1 = No Response Error asserted

26	COLL0	RWC	0x0	Collision Error Channel 0: This register indicates data collision between controller and main unit. Once set this bit remains set until it is cleared by the main processor. To clear this bit write '1' to this register. Write: 0 = No effect 1 = Clear Collision Error Read: 0 = Collision Error not asserted 1 = Collision Error asserted
25	OVRUN0	RWC	0x0	Over Run Error Channel 0: This register indicates that the main unit has received more data then expected. Once set this bit remains set until it is cleared by the main processor. To clear this bit write '1' to this register. Write: 0 = No effect 1 = Clear Over Run Error Read: 0 = Over Run Error not asserted 1 = Over Run Error asserted
24	UNRUN	RWC	0x0	Under Run Error Channel 0: This register indicates that the main unit has received less data then expected. Once set this bit remain set until it is cleared by the main processor. To clear this bit write '1' to this register. Write: 0 = No effect 1 = Clear Under Run Error Read: 0 = Under Run not asserted 1 = Under Run asserted
23..22		R	0x0	Reserved
21	RDST1	R	0x0	Read Status SIC1OINBUF Register: See SISR[RDST0].
20	WRST1	R	0x0	Write Status SIC0OUTBUF Register: See SISR[WRST0].
19	NOREP1	RWC	0x0	No Response Error Channel 1: See SISR[NOREP0].
18	COLL1	RWC	0x0	Collision Error Channel 1: See SISR[COLL0].
17	OVRUN1	RWC	0x0	Over Run Error Channel 1: See SISR[OVRUN0].
16	UNRUN1	RWC	0x0	Under Run Error Channel 1: See SISR[UNRUN0].
15..14		R	0x0	Reserved
13	RDST2	R	0x0	Read Status SIC1OINBUF Register: See SISR[RDST2].

12	WRST2	R	0x0	Write Status SIC0OUTBUF Register: See SISR[WRST2].
11	NOREP2	RWC	0x0	No Response Error Channel 2: See SISR[NOREP0].
10	COLL2	RWC	0x0	Collision Error Channel 2: See SISR[COLL0].
9	OVRUN2	RWC	0x0	Over Run Error Channel 2: See SISR[OVRUN0].
8	UNRUN2	RWC	0x0	Under Run Error Channel 2: See SISR[UNRUN0].
7..6		R	0x0	Reserved
5	RDST3	R	0x0	Read Status SIC1OINBUF Register: See SISR[RDST2].
4	WRST3	R	0x0	Write Status SIC0OUTBUF Register: See SISR[WRST2].
3	NOREP3	RWC	0x0	No Response Error Channel 3: See SISR[NOREP0].
2	COLL3	RWC	0x0	Collision Error Channel 3: See SISR[COLL0].
1	OVRUN3	RWC	0x0	Over Run Error Channel 3: See SISR[OVRUN0].
0	UNRUN3	RWC	0x0	Under Run Error Channel 3: See SISR[UNRUN0].

SIEXILK SI EXI Clock Lock

Mnemonic: SIEXILK
Offset: 0x3C
Size 32 bits

SIEXILK				
Bits	**Mnemonic**	**Type**	**Reset**	**Description**
31	LOCK	RW	0x1	Lock: This bit prevents the main processor from setting the EXI clock frequencies to 32 MHz. 0 = EXI Clocks Unlocked, 32MHz EXICLK setting permitted. 1 = EXI Clock Locked, 32MHz EXICLK setting not permitted.
30..0		R	0x0	Reserved

Figure 8 is an even more detailed overall view of serial interface 1000 showing the details of serial interface communication circuitry and registers 1012. Controllers 52a and 52b (and 52c and 52d, if present) are connected to game console 54 via connector ports 1002. Modem 1404 modulates and demodulates data transferred between the controllers and the console. In the example system, communication between the console and the controllers uses duty-cycle (pulse-width) modulation and the data is communicated over one line. The

communication is half-duplex. The byte transfer order is "big-endian" in which within a given multi-byte numeric representation, the most significant byte has the lowest address (i.e., the data is transferred "big-end" first). Controller input/output buffer 1016 is used for normal data transfers involving controllers 52a-52d. As shown in Figure 8 and as will be explained in greater detail below, input/output buffer 1016 is arranged as a double buffer. Communication RAM 1014 is provided for use in variable-size data transfers to and from controllers 52a-52d. In the example system, the maximum data size of these variable-size data transfers is 32 words. Of course, the present invention is not limited in this respect. Channel selector circuit 1408 controls selectors 1412a-1412d to selectively connect modem 1404 to either communication RAM 1014 or input/output buffer 1016. An HV counter latch circuit 1406 latches the screen position of a flash signal when a trigger input is received from a light gun unit. In the example system shown in Figure 8, triggers inputs to the HV counter latch circuit 1406 are provided for connectors 1 and 2 only. It will be apparent that trigger inputs may be provided for the other connectors if desired. HV counter latch circuit 1406 may also be used with light pens connected to connectors 1 and/or 2.

Figure 9 shows additional details of the connections between controllers 52a-52d and serial interface 1000. The outputs of modem 1404 are supplied to controllers 52a-52d through open-drain driver buffers 1430 and the inputs from controllers 52a-52d are supplied to modem 1404 through Schmitt buffers 1432. Similarly, the outputs of controllers 52a-52d are supplied to serial interface 1000 through driver buffers 1434. The outputs to controllers 52a-52d from serial interface 1000 are supplied to controller integrated circuits (ICs) 1436.

Figure 10 shows details of one of controllers 52. Controller 52 is connected to serial interface 1000 via a cable (not shown) about 2 meters in length in the

example system. In other implementations, the communication between the controllers and the console may be over a wireless communication path such as infrared or radio frequency. Controller 52 contains an interface circuit 1504 arranged between the controller components and serial interface 1000. A button data input circuit 1502 accepts inputs from buttons disposed on an external surface of controller 52. "Buttons" as used herein refers to any device that is manipulable by a user to cause a game or other application to start, characters to move, etc. and includes, for example, buttons, switches, joysticks, and the like. The length of the controller button and status data that is provided to serial interface 1000 is 64 bits. Vibration circuit 1506 is responsive to signals from main unit 54 via serial interface 1000 for selectively vibrating the housing of controller 52 to provide sensations to the a game player. Controller 52 also includes a 64 kbit EEPROM 1510 with a unique identifier number that is usable, for example, to store game back-up data, high game scores, etc. Controller 52 also includes a motor control system 1506 for controlling an external motor and two output ports (not shown) connectable to external components such as a motor.

Polling Via Serial Interface 1000

The input/output data size is fixed in the standard command/response packet in the example implementation. This packet is used primarily for reading data of the controllers. This data includes data indicative of which buttons are being pressed, is there a controller plugged in, the value of the analog joysticks, etc. This data is used as player input for a game (e.g., move character left/right, look up/down, fire gun, etc.). Because the polling of the controllers is performed by serial interface 1000, the workload of main processor 110 is reduced. Thus, for example, the last state of the controller 52 is in the double-buffered processor

input/output register 1016 so that main processor 110 can simply read the register to determine the controller state.

An output command packet includes a 1-byte command and 2 bytes of data. With reference to the serial interface register map set forth above, SIC0OUTBUF for sending a command packet to a controller connected to connector port 1002(0), for example, includes a command byte register CMD that is an opcode for the command sent to the controller during each command/response packet. OUTPUT0 and OUTPUT1 registers are for first and second data bytes of the command packet. An input response packet includes 8 bytes of data, of which 62 bits are button data and 2 bits are controller status bits. Again with reference to the serial interface register map, SIC0INBUF (SI Channel 0 Input Buffer High) for receiving the response packet from the controller includes ERRSTAT, ERRLATCH, INPUT0, INPUT1, INPUT2 and INPUT3 registers. The ERRSTATUS bit represents the current error status for the last polling transfer on channel 0. This register is updated after each polling transfer on this channel. The ERRLATCH bit is an error status summary of the error bits for channel 0. If an error has occurred on a past transfer on channel 0 (polling or a transfer from communication RAM 1014), this bit will be set. To determine the exact error, the SISR register (see register map) may be read. The ERRLATCH bit is actually an "or" of the latched error status bits for channel 0 in the SISR. The bit is cleared by clearing the appropriate error status bits latched in the SISR. The no response error indicates that a controller is not present on the channel. INPUT0, INPUT1, INPUT2 and INPUT3 are first through fourth data bytes of the response packet sent from controller 52 to serial interface 1000 for channel 0. In the example, implementation, the most significant two bits of INPUT0 are assumed to be "0", so they are not included in

INPUT0. The fifth through eighth data bytes of the response packet are in the register SIC0INBUF (SI Channel 0 Input Buffer Low).

The characteristics of the polling of the controllers are determined in accordance with registers in SIPOLL (SI Poll Register). With reference to the register map, SIPOLL includes an X lines register and a Y times register. The X lines register determines the number of horizontal video lines between polling ("the polling interval"). The polling interval begins at the beginning of vertical blanking as shown in Figure 12. 0x07 is the minimum setting for the X lines register as determined by the time required to complete a single polling of the controller in the example system. The maximum setting depends on the current video mode (the number of lines between vertical sync signals) and the Y times register. Of course, the invention is not limited to any particular maximum or minimum settings. The Y times register determines the number of times the controllers are polled in a single frame. This is also shown in Figure 12.

Thus, in the controller polling process, main processor 110 writes a command to one or more of the output buffers and sets the X line and Y times registers. When main processor 110 sets one or more enable registers EN0, EN1, EN2 and EN3 of the SI Poll Register to "1", continuous polling of the corresponding controllers is started. The polling of the enabled channels is started at the same time and serial interface 1000 starts to poll at the beginning of vertical blanking and polling is repeated Y times per one frame. As noted above, the interval between pollings is determined in accordance with the setting of the X line register and the relationship between X line interval and Y times per frame is illustrated in Figure 12. When main processor 110 sets the X line register, the Y times register and the enable register(s), controller access is started at the next vertical blanking. In the example implementation, the default setting for the Y

times register is "0". Therefore, main processor 110 must set this register to use the controllers. To stop polling of a particular controller, its enable bit is set to 0; polling is stopped immediately after the current transaction.

As mentioned above, input/output buffer 1016 is actually a double buffer. The output buffer is comprised of 3 bytes for each of the four channels as described above. With reference to Figure 8, the output data is copied from output buffer 1 to output buffer 0 of the double buffer when main processor 110 finishes writing to output buffer 1. A WR register of SISR (SI Status Register) controls and indicates whether the SICnOUTBUFs have been copied to the double buffered output buffers. The bit is set to "1" when the buffer is not yet copied and is cleared to "0" when copying is completed. Thus, if "0" is returned when main processor 110 reads WR, the copy operation is complete. If "1" is returned, the copy operation is not complete. Thus, this register must be "0" before main processor 110 writes to buffer 1. While data is transmitted from buffer 0 to the controller, copy from buffer 1 to buffer 0 is locked.

The input buffer is 8 bytes for each of the four channels and input data is copied from input buffer 0 to input buffer 1 after the controller data is received. Main processor 110 reads input buffer 1 directly per 32 bits. If new data were to be written to the low 32 bits while main processor 110 is reading the high 32 bits, the data may be incorrect. Accordingly, when main processor 110 is reading the upper 32 bits, input buffer 1 is locked. When main processor 110 starts to read the lower 32 bits, buffer 1 is unlocked.

Ordinarily, output data is copied from buffer 1 to buffer 0 immediately after it is written to output buffer 1 by main processor 110. However, the copying may also be timed to start with vertical blanking in order to control the timing of commands to 3D LCD shutter glasses connected to display controller 162. To

enable such timing, VBCPY may be set to "1" so that copies to buffer 0 occur only upon vertical blanking. The default value for the VBCPY bits is "0".

The size of the data involved in the polling is fixed. However, there are occasions on which it is desirable to transfer larger amounts of data between main console 54 and controllers 52 (e.g., writing game data to a nonvolatile memory built-in or attached to controller 52). Example serial interface 100 provides the capability of transferring larger amounts of data in accordance with a process that is explained with reference to Figure 11. SICOMCSR (SI Communication Control Status Register) includes a register OUTLNGTH for setting the communication channel output length (in bytes); INLNGTH for setting the communication channel input length (in bytes); and CHANNEL for determining which serial interface channel will be used. The minimum transfer that may be designated by OUTLNGTH and INLNGTH is 1 byte. A value of 0x00 will transfer 128 bytes. To transfer data, command and output data is set to communication RAM 1410 and main processor 110 sets the OUTLNGTH, INLNGTH and CHANNEL registers. The setting ("1") of the TSTART bit register causes execution of the current communication transfer. The transfer begins immediately after the current transaction on the designated channel is completed. When read, the TSTART bit represents the current transfer status. Once a communication transfer is executed, polling resumes at the next vertical blanking interval if the channels EN bit is set.

When main processor 110 sets the TSTART bit to "1", the transfer mode of the channel designated in the CHANNEL register is changed to Communication RAM mode by selector circuit 1408 (see Figure 8) which controls the appropriate selector circuit to connect that channel to communication RAM 1410. In the example of Figure 11, channel 1 is the designated channel. Output data is then transmitted from communication RAM 1410 and is output to controller 52 via

modem 1404. Controller 52 may also transmit input data to channel 1 and this data is communicated to communication RAM 1410 via selector 1412a. The input data is pushed from communication RAM 1410 to main processor 110. Modem 1404 automatically clears the TSTART bit when the transfer is complete.

During Communication RAM transfer mode on channel 1, the remaining channels (i.e., channels 2, 3 and 4) continue to transfer data using input/output buffer 1414. During communication RAM transfers, main processor 110 generates an Addr CRC and transmits the CRC to the controller. The controller processes the Addr CRC to check that the data has been correctly received. In the case of a communication RAM transfer, the controller returns Data CRC. The main processor 110 generates a data CRC and compares this with the controller returned Data CRC.

Figure 13 illustrates an example light gun connectable to connector port 1002(0) or 1002(1) of main console 54. Various types of light guns may be used and two of these types are discussed below. A first type detects a target point within one frame and generally requires that the screen and/or the target point be a bright color. A second type requires two frames to detect the target point because main processor 110 flashes the screen after detection of the trigger button and starts to detect the target point at the next frame. This second type of gun typically does not require the screen to be as bright as is needed for guns of the first type. HV counter latch circuit 1406 is usable in connection with either type of light gun. The light gun includes a gun trigger 1702. If gun trigger 1702 is pulled, a trigger signal input is transmitted for one frame. A flash signal input passing through lens 1720 is detected by a photodetector 1722, amplified by an amplifier 1724 and supplied to logic circuit 1706 as the flash signal. Interface circuit 1704 controls whether the flash signal is passed via logic gate 1706 by controlling the signal

level on the flash signal control line. If the flash signal control line is high, the flash signal is passed to HV counter latch circuit as a counter latch signal. Interface circuit 1704 controls whether the trigger signal is passed. A trigger control switch 1708 is selectively connected to power source 1710 or the trigger signal line.

In the first type of light gun, as soon as trigger 1702 is pulled, the gun starts to detect the flash signal and, if the flash signal is detected, the counter latch signal is transmitted. In the second type of light gun, when trigger 1702 is pulled, main processor 110 starts to flash the screen at the next frame. The flash signal is detected during this frame and the counter latch signal is transmitted. With a light pen, the flash signal is detected per frame and the counter latch signal is transmitted per frame.

A Gun Trigger Mode register determines the gun trigger mode and enables the counter latch register for each of channels 1 and 2. An "always" mode is set so that the gun detects the flash signal per frame. This mode may be set for the first type of light gun and the light pen. A "1 frame" and "2 frame" mode is set so that the gun detects a flash signal for 1 or 2 frames. These modes may be set for the second type of light gun.

The HV latch counter circuit receives a counter latch signal transmitted from the light gun. In this case, the position of the flash signal is latched to the HV counter latch register. The register is shared by the 2 channels.

HV counter latch	32 bits (R)
Horizontal count	[10..0] bits
Vertical count	[20..12] bits
Field	bit 24

Counter latch signal bit 31

Figure 14A depicts the output timing of the counter latch signal for the light gun of the first type. As noted above, the screen should preferably be bright for this type of light gun because the gun needs to detect a flash signal at any time. For this gun type, the trigger signal controller is always connected to the trigger side. When the trigger is pulled, flash signal detection is started. If the flash signal is detected within one frame, the counter latch signal is transmitted and the counter latch register gets the position of the flash signal. The action that results from pulling the trigger is drawn at the next frame.

Figure 14B depicts the output timing of the counter latch signal for the light gun of the second type. For this gun type, the trigger signal controller is always connected to the power side. When the trigger is pulled, the screen is a bright color at the next frame and flash signal detection is started. When the flash signal is detected, the counter latch signal is transmitted and the counter latch register gets the position of the flash signal. The action that results from pulling the trigger is drawn at the next frame.

Figure 14C depicts the output timing of the counter latch signal for the light pen. The screen is preferably bright for this type of light gun because the gun needs to detect a flash signal at any time. The trigger signal controller is always connected to the power side and the flash signal control line is always set to 1. The flash signal is detected per frame. The counter latch signal is transmitted and the counter latch register gets the position of the flash signal. The action that results from touching the pen on the screen is drawn at the next frame.

Example Signal Form

The signal form of the data is made up of a separation signal, a bit signal, an end signal and a continue signal.

Figure 15(A) depicts a separation signal that is used to distinguish the end of transmission. The separation signal is transmitted before the transmitter (main unit 54) transmits a command and the receiver (controller 52) distinguishes the start of the command. Thus, it is important that the receiver distinguishes this signal.

Figures 15(B) and 15(C) depict bit signals that are transmitted using the DATA line. The bit signals are duty-cycle (pulse-width) modulated as shown in Figures 15(B) and 15(C).

Figure 15(D) depicts an end signal for indicating the end of the bit signal transmission. The bit signal ends by falling from high level to low level so the data line is still at low level. However, to release the data line, the transmitter must return the data line to high level. The end signal is used for this purpose.

Figure 15(E) depicts a continue signal used to change the direction of transmission. The transmitter returns the data line to high level and next the transmitter starts to transmits the Bit signal after a few intervals. The interval between the End signal and the Bit signal is the Continue signal. This signal is controlled by the next transmitter and is used when the receiver replies to a command of the transmitter.

Example Communication Protocol

Figures 16A and 16B depict frame units in which communication over the communication path occurs. The transmitter transmits the Command frame shown in Figure 16A and the receiver responds to the transmitter with a Response frame shown in Figure 16B. As shown in Figure 16A, the Command frame includes a

separation signal, data and an end signal. As shown in Figure 16B, the Response frame includes a continue signal, data and an end signal. The data is 8-bit aligned.

Example Channel Lines

Figure 17 depicts the lines for the channels of serial interface 1000. A GND line is connected to a ground terminal and a PWR line connected to a VDD terminal is used to supply power from main console 54 to controller 52. The DATA line is used to transfer data between main console 54 and controller 52. DATA line is bi-directional and half-duplex. DATA line includes a wired-OR circuit and the transmitter pulls it to High level. The receiver may also pull it to High level. When the transmitter transmits signals, it pulls the DATA line to low level for certain times. The mechanical GND line and the mechanical PWR line are used when power is supplied to an external device coupled to the controller such as a motor. Finally, the counter latch line is used to transfer the flash signal detected from a light gun, a write pen, etc.

Other Example Compatible Implementations

Certain of the above-described system components 50 could be implemented as other than the home video game console configuration described above. For example, one could run graphics application or other software written for system 50 on a platform with a different configuration that emulates system 50 or is otherwise compatible with it. If the other platform can successfully emulate, simulate and/or provide some or all of the hardware and software resources of system 50, then the other platform will be able to successfully execute the software.

As one example, an emulator may provide a hardware and/or software configuration (platform) that is different from the hardware and/or software

configuration (platform) of system 50. The emulator system might include software and/or hardware components that emulate or simulate some or all of hardware and/or software components of the system for which the application software was written. For example, the emulator system could comprise a general purpose digital computer such as a personal computer, which executes a software emulator program that simulates the hardware and/or firmware of system 50.

Some general purpose digital computers (e.g., IBM or MacIntosh personal computers and compatibles) are now equipped with 3D graphics cards that provide 3D graphics pipelines compliant with DirectX or other standard 3D graphics command APIs. They may also be equipped with stereophonic sound cards that provide high quality stereophonic sound based on a standard set of sound commands. Such multimedia-hardware-equipped personal computers running emulator software may have sufficient performance to approximate the graphics and sound performance of system 50. Emulator software controls the hardware resources on the personal computer platform to simulate the processing, 3D graphics, sound, peripheral and other capabilities of the home video game console platform for which the game programmer wrote the game software.

Figure 18A illustrates an example overall emulation process using a host platform 1201, an emulator component 1303, and a game software executable binary image provided on a storage medium 62. Host 1201 may be a general or special purpose digital computing device such as, for example, a personal computer, a video game console, or any other platform with sufficient computing power. Emulator 1303 may be software and/or hardware that runs on host platform 1201, and provides a real-time conversion of commands, data and other information from storage medium 62 into a form that can be processed by host 1201. For example, emulator 1303 fetches "source" binary-image program

instructions intended for execution by system 50 from storage medium 62 and converts these program instructions to a target format that can be executed or otherwise processed by host 1201.

As one example, in the case where the software is written for execution on a platform using an IBM PowerPC or other specific processor and the host 1201 is a personal computer using a different (e.g., Intel) processor, emulator 1303 fetches one or a sequence of binary-image program instructions from storage medium 62 and converts these program instructions to one or more equivalent Intel binary-image program instructions. The emulator 1303 also fetches and/or generates graphics commands and audio commands intended for processing by the graphics and audio processor 114, and converts these commands into a format or formats that can be processed by hardware and/or software graphics and audio processing resources available on host 1201. As one example, emulator 1303 may convert these commands into commands that can be processed by specific graphics and/or or sound hardware of the host 1201 (e.g., using standard DirectX, OpenGL and/or sound APIs).

An emulator 1303 used to provide some or all of the features of the video game system described above may also be provided with a graphic user interface (GUI) that simplifies or automates the selection of various options and screen modes for games run using the emulator. In one example, such an emulator 1303 may further include enhanced functionality as compared with the host platform for which the software was originally intended.

Figure 18B illustrates an emulation host system 1201 suitable for use with emulator 1303. System 1201 includes a processing unit 1203 and a system memory 1205. A system bus 1207 couples various system components including system memory 1205 to processing unit 1203. System bus 1207 may be any of

several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. System memory 1207 includes read only memory (ROM) 1252 and random access memory (RAM) 1254. A basic input/output system (BIOS) 1256, containing the basic routines that help to transfer information between elements within personal computer system 1201, such as during start-up, is stored in the ROM 1252. System 1201 further includes various drives and associated computer-readable media. A hard disk drive 1209 reads from and writes to a (typically fixed) magnetic hard disk 1211. An additional (possible optional) magnetic disk drive 1213 reads from and writes to a removable "floppy" or other magnetic disk 1215. An optical disk drive 1217 reads from and, in some configurations, writes to a removable optical disk 1219 such as a CD ROM or other optical media. Hard disk drive 1209 and optical disk drive 1217 are connected to system bus 1207 by a hard disk drive interface 1221 and an optical drive interface 1225, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules, game programs and other data for personal computer system 1201. In other configurations, other types of computer-readable media that can store data that is accessible by a computer (e.g., magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROMs) and the like) may also be used.

A number of program modules including emulator 1303 may be stored on the hard disk 1211, removable magnetic disk 1215, optical disk 1219 and/or the ROM 1252 and/or the RAM 1254 of system memory 1205. Such program modules may include an operating system providing graphics and sound APIs, one or more application programs, other program modules, program data and game

data. A user may enter commands and information into personal computer system 1201 through input devices such as a keyboard 1227, pointing device 1229, microphones, joysticks, game controllers, satellite dishes, scanners, or the like. These and other input devices can be connected to processing unit 1203 through a serial port interface 1231 that is coupled to system bus 1207, but may be connected by other interfaces, such as a parallel port, game port Fire wire bus or a universal serial bus (USB). A monitor 1233 or other type of display device is also connected to system bus 1207 via an interface, such as a video adapter 1235.

System 1201 may also include a modem 1154 or other network interface means for establishing communications over a network 1152 such as the Internet. Modem 1154, which may be internal or external, is connected to system bus 123 via serial port interface 1231. A network interface 1156 may also be provided for allowing system 1201 to communicate with a remote computing device 1150 (e.g., another system 1201) via a local area network 1158 (or such communication may be via wide area network 1152 or other communications path such as dial-up or other communications means). System 1201 will typically include other peripheral output devices, such as printers and other standard peripheral devices.

In one example, video adapter 1235 may include a 3D graphics pipeline chip set providing fast 3D graphics rendering in response to 3D graphics commands issued based on a standard 3D graphics application programmer interface such as Microsoft's DirectX 7.0 or other version. A set of stereo loudspeakers 1237 is also connected to system bus 1207 via a sound generating interface such as a conventional "sound card" providing hardware and embedded software support for generating high quality stereophonic sound based on sound commands provided by bus 1207. These hardware capabilities allow system 1201 to provide sufficient

graphics and sound speed performance to play software stored in storage medium 62.

While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiment, it is to be understood that the invention is not to be limited to the disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the scope of the appended claims.